

PRACTICAL NO. 1 Implementation of Remote Procedure Call

1. To implement a Server calculator using RPC concept. (Make use of datagram)

Code:

CalculatorServer.java

```

package udpcalc;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class CalculatorServer {
    public static void main(String[] args) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(5000);
            byte[] receiveBuffer = new byte[1024];
            System.out.println("UDP Calculator Server is running on port 5000...");
            while (true) {
                DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
                serverSocket.receive(receivePacket);
                String request = new String(receivePacket.getData(), 0, receivePacket.getLength());
                System.out.println("Received: " + request);
                // Parse request: "add:5:3"
                String[] parts = request.trim().split(":");
                String operation = parts[0].toLowerCase();
                double num1 = Double.parseDouble(parts[1]);
                double num2 = Double.parseDouble(parts[2]);
                double result = 0;
                switch (operation) {
                    case "add": result = num1 + num2; break;
                    case "sub": result = num1 - num2; break;
                    case "mul": result = num1 * num2; break;
                    case "div": result = (num2 != 0) ? num1 / num2 : Double.NaN; break;
                    default: result = Double.NaN;
                }
                String response = "Result:" + result;
                byte[] sendBuffer = response.getBytes();
                DatagramPacket sendPacket = new DatagramPacket(
                    sendBuffer,
                    sendBuffer.length,
                    receivePacket.getAddress(),
                    receivePacket.getPort()
                );
                serverSocket.send(sendPacket);
                System.out.println("Sent to client: " + response);
            }
        }
    }
}

```

```
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

CalculatorClinet.java

```
package udpcalc;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class CalculatorClient {
    public static void main(String[] args) {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter operation (add/sub/mul/div): ");
            String op = sc.next();
            System.out.print("Enter first number: ");
            double num1 = sc.nextDouble();
            System.out.print("Enter second number: ");
            double num2 = sc.nextDouble();
            String request = op + ":" + num1 + ":" + num2;
            byte[] sendBuffer = request.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, serverAddress,
5000);
            clientSocket.send(sendPacket);
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
            clientSocket.receive(receivePacket);
            String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Response from server: " + response);
            clientSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
UDP Calculator Server is running on port 5000...
```

```
Enter operation (add/sub/mul/div): add
Enter first number: 10
Enter second number: 2
Response from server: Result:12.0
```

2. To implement a Date Time Server using RPC concept. (Make use of datagram)**Code:****DateTimeServer.java**

```
package udpDateTime;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
public class DateTimeServer {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        try {
            DatagramSocket serverSocket = new DatagramSocket(5000);
            byte[] receiveBuffer = new byte[1024];
            System.out.println("UDP date time server is running on port 5000...");
            while(true)
            {
                DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
                receiveBuffer.length);
                serverSocket.receive(receivePacket);
                String cureentDateTime =
                LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss"));
                byte[] sendBuffer = cureentDateTime.getBytes();
                InetAddress clientAddress = receivePacket.getAddress();
                int clientPort = receivePacket.getPort();
                DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
                sendBuffer.length,clientAddress ,clientPort);
```

```

        serverSocket.send(sendPacket);
        System.out.println("sent date/time to client:
"+clientAddress+":"+clientPort);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

DateTimeClient.java
package udpDateTime;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
public class DateTimeClient {
    public static void main(String[] args) {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            byte[] sendBuffer = "getDateTime".getBytes();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer,
sendBuffer.length,serverAddress,5000);
            clientSocket.send(sendPacket);
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer,
receiveBuffer.length);
            clientSocket.receive(receivePacket);
            String dateTime = new
String(receivePacket.getData(),0,receivePacket.getLength());
            System.out.println("Received from server : "+dateTime);
            clientSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Output:

UDP Calculator Server is running on port 5000...

```
UDP date time server is running on port 5000...
sent date/time to client: /127.0.0.1:50699
```

```
|Received from server : 03-09-2025 09:48:11
```

3. To implement a Server calculator using RPC concept. (Make use of Server Socket)

Code:

CalculatorServer.java

```
package tcpalc;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
public class CalculatorServer {
    public static void main(String[] args)
    {
        try (ServerSocket serverSocket = new ServerSocket(1235)) {
            System.out.println("Calculator Server is running on port 1235...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client Connected!");

                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                String request = in.readLine();
                System.out.println("Request: " + request);

                String[] parts = request.split(":");
                String operation = parts[0].toLowerCase();
                double num1 = Double.parseDouble(parts[1]);
                double num2 = Double.parseDouble(parts[2]);
                double result;
                switch (operation) {
                    case "add": result = num1 + num2; break;
                    case "sub": result = num1 - num2; break;
                    case "mul": result = num1 * num2; break;
                    case "div": result = (num2 != 0) ? num1 / num2 : Double.NaN; break;
                }
                out.println(result);
            }
        }
    }
}
```

```
        default: result = Double.NaN;  
    }  
    out.println("Result:" + result);  
    clientSocket.close();  
}  
}  
}  
}  
}
```

CalculatorClient.java

```
CalculatorClient.java
package tcpcalc;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
public class CalculatorClient {
    public static void main(String[] args) {
        try (Socket s = new Socket("localhost", 1235)) {
            BufferedReader in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
            PrintWriter out = new PrintWriter(s.getOutputStream(), true);
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter an operation (add/sub/mul/div):");
            String operation = sc.next();
            System.out.println("Enter number 1:");
            double num1 = sc.nextDouble();
            System.out.println("Enter number 2:");
            double num2 = sc.nextDouble();
            String request = operation + ":" + num1 + ":" + num2;
            out.println(request);
            String response = in.readLine();
            System.out.println("Response from server = " + response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

Calculator Server is running on port 1235...

```

terminated> CalculatorClient(1,[javaapplication1],21876)
Enter an operation (add/sub/mul/div):
add
Enter number 1:
5
Enter number 2:
9
Response from server = Result:14.0

```

4. To implement a Date Time Server using RPC concept. (Make use of Server Socket)

Code:

DateTimeServer.java

```

package tcpDateTime;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
public class DateTimeServer {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(5001)) {
            System.out.println("Date Time TCP Server is running on port 5000...");
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client Connected!");
                BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true); // auto-flush
                String request = in.readLine(); // expecting "getDateTime"
                System.out.println("Request received: " + request);
                if ("getDateTime".equalsIgnoreCase(request)) {
                    String currentDateTime = LocalDateTime.now()
                        .format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss"));
                    out.println("Current Date & Time: " + currentDateTime);
                } else {
                    out.println("Invalid Request");
                }
                clientSocket.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

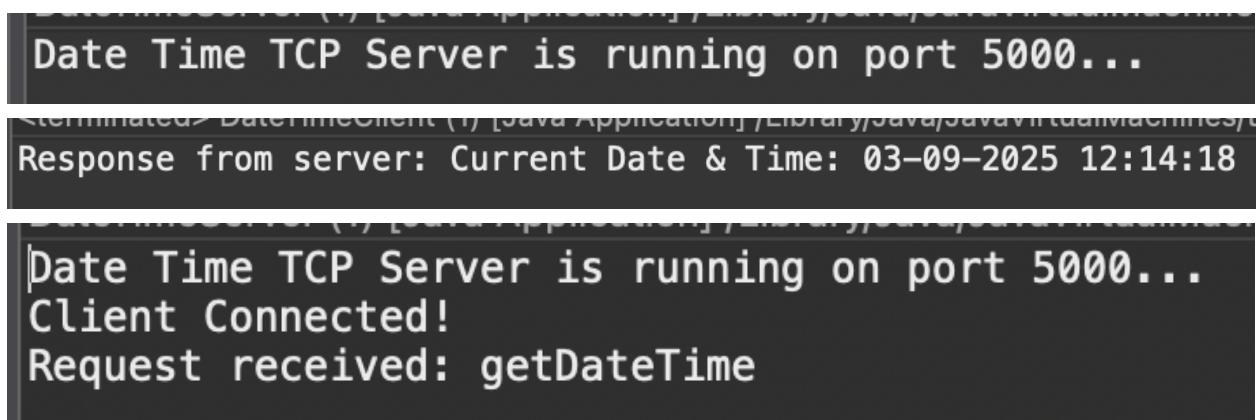
```

```

        }
    }
}

DateTimeClient.java
package tcpDateTime;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
public class DateTimeClient {
    public static void main(String[] args) {
        try (Socket s = new Socket("localhost", 5001)) {
            BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream()));
            PrintWriter out = new PrintWriter(s.getOutputStream(), true); // auto-flush
            // Send RPC request
            out.println("getDateTime");
            // Receive RPC response
            String response = in.readLine();
            System.out.println("Response from server: " + response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Output:


The screenshot shows two terminal windows. The top window is titled 'Date Time TCP Server' and displays the message 'Date Time TCP Server is running on port 5000...'. The bottom window is titled 'DateTimeClient' and shows the following interaction:

```

Date Time TCP Server is running on port 5000...
<terminated> Date Time Client (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java -jar Date Time Client.jar
Response from server: Current Date & Time: 03-09-2025 12:14:18

Date Time TCP Server (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-17.0.2.jdk/Contents/Home/bin/java -jar Date Time Server.jar
Date Time TCP Server is running on port 5000...
Client Connected!
Request received: getDateTime

```

5. To implement Equation solver using Datagram. The client should provide an equation to the Server through an interface. The server will solve the expression given by the client. $(a-b)^2 = a^2 - 2ab + b^2$; If $a = 5$ and $b = 2$ then return value = $5^2 - 2 \cdot 5 \cdot 2 + 2^2 = 9$.

code:

EquationServer.java

```
package udpEquation;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class EquationServer {
    public static void main(String[] args) {
        try {
            DatagramSocket serverSocket = new DatagramSocket(5002);
            byte[] receiveBuffer = new byte[1024];
            System.out.println("UDP Equation Solver Server is running on port 5002... ");
            while (true) {
                // Receive request from client
                DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
                serverSocket.receive(receivePacket);
                String request = new String(receivePacket.getData(), 0, receivePacket.getLength());
                System.out.println("Received: " + request);
                // Request format: "a:b" (values for a and b)
                String[] parts = request.trim().split(":");
                double a = Double.parseDouble(parts[0]);
                double b = Double.parseDouble(parts[1]);
                // Solve  $(a-b)^2 = a^2 - 2ab + b^2$ 
                double result = (a * a) - (2 * a * b) + (b * b);
                String response = "Result of  $(a-b)^2 =$ " + result;
                byte[] sendBuffer = response.getBytes();
                // Send result back to client
                DatagramPacket sendPacket = new DatagramPacket(
                    sendBuffer, sendBuffer.length,
                    receivePacket.getAddress(), receivePacket.getPort()
                );
                serverSocket.send(sendPacket);
                System.out.println("Sent: " + response);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

EquationClient.java

```

package udpEquation;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class EquationClient {
    public static void main(String[] args) {
        try {
            DatagramSocket clientSocket = new DatagramSocket();
            InetAddress serverAddress = InetAddress.getByName("localhost");
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter value of a: ");
            double a = sc.nextDouble();
            System.out.print("Enter value of b: ");
            double b = sc.nextDouble();
            String request = a + ":" + b;
            byte[] sendBuffer = request.getBytes();
            // Send request to server
            DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, serverAddress,
5002);
            clientSocket.send(sendPacket);
            // Receive response from server
            byte[] receiveBuffer = new byte[1024];
            DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);
            clientSocket.receive(receivePacket);
            String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Response from server: " + response);
            clientSocket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Output:

UDP Equation Solver Server is running on port 5002...

```
UDP Equation Solver Server is running on port 5002...
Received: 5.0:2.0
Sent: Result of (a-b)^2 = 9.0
```

```
Enter value of a: 5
Enter value of b: 2
Response from server: Result of (a-b)^2 = 9.0
```

Conclusion:

This practical demonstrated implementing Remote Procedure Call (RPC) using both UDP and TCP. We developed calculator, date-time, and equation solver programs where clients invoked remote functions as if they were local. The exercises improved understanding of socket programming, data marshalling/unmarshalling, and client-server communication in distributed applications.